Reducing Communication Overhead for Average Consensus

Mahmoud El Chamie, Giovanni Neglia, Konstantin Avrachenkov

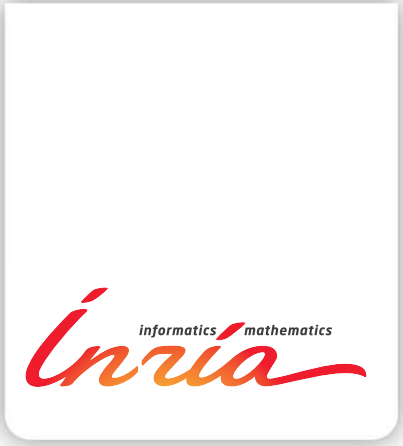
RESEARCH REPORT

N° 8025

July 2012

Project-Teams Maestro

ISSN 0249-6399 ISRN INRIA/RR--8025--FR+ENG



Reducing Communication Overhead for Average Consensus

Mahmoud El Chamie*, Giovanni Neglia†, Konstantin Avrachenkov‡

Project-Teams Maestro

Research Report n° 8025 — July 2012 — 22 pages

Abstract: Average consensus is an iterative protocol where nodes in a network, each having an initial scalar value called estimate, perform a distributed algorithm to calculate the average of all estimates presented in the network by using only local communication. With every iteration, nodes receive the estimates from their neighbors, and they update their own estimate by the weighted average of the received ones. While the average consensus protocol converges asymptotically to consensus, implementing a termination algorithm is challenging when nodes are not aware of some global information (e.g. the diameter of the network or the number of nodes presented). In this report, we are interested in decreasing the rate of the messages sent in the network as the estimates are closer to consensus. We propose a totally distributed algorithm for average consensus where nodes send more messages when the nodes have large differences in their estimates, and reduce their rate of sending messages when the consensus is almost reached. The convergence of the system is guaranteed to be within a predefined margin ϵ from the true average and the communication overhead is largely reduced.

Key-words: average consensus, energy reduction, termination protocol, distributed algorithms

* Inria Sophia Antipolis, France, Mahmoud.El_Chamie@inria.fr
† Inria Sophia Antipolis, France, Giovanni.Neglia@inria.fr
‡ Inria Sophia Antipolis, France, K.Avrachenkov@sophia.inria.fr

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

La Réduction de Charge de Communication pour le Consensus de Moyenne

Résumé : Le consensus de moyenne est un protocole itératif où les nœuds d'un réseau, ayant chacun une estimation initiale, exécutent un algorithme distribué pour calculer la moyenne de ces estimations en utilisant uniquement les communication locales. A chaque itération, les nœuds échangent leurs estimations avec leurs voisins. Ces estimations seront remplacées par la moyenne pondérée de celles reçues. La convergence du consensus de moyenne est asymptotique et la mise en œuvre d'un protocole de terminaison est difficile lorsque les nœuds ne connaissent pas l'estimation global (par exemple, le diamètre du réseau ou le nombre de nœuds). Dans ce rapport, nous intéressons à la réduction du taux de messages envoyés dans le réseau quand les estimations deviennent proche du consensus. Nous présentons un algorithme de consensus de moyenne totalement distribué, où les nœuds envoient plus de messages lorsque la différence entre leurs estimations est grande et moins de messages lorsque le système est à peu près convergeant. La convergence du système est garantie d'être proche de la vraie moyenne et le coût des communications est fortement réduit.

Mots-clés : consensus de moyenne, réduction de l'énergie, protocole de terminaison, algorithme distribué

1 Introduction

Average consensus protocols are used to find the average across initial measurements presented at nodes in a network in a distributed manner having no central entity to control and monitor the network. This problem is gaining interest nowadays due to its wide domain of applications as in cooperative robot motions [1], resource allocation [2], and environmental monitoring [3], in addition to the existence of large networks such as wireless sensor networks for which such consensus algorithms are necessary [4]. Under this decentralized approach, nodes select a weight for the estimate of every neighbor and perform a weighted average of the values in their closed neighborhood. By considering special conditions on the selected weights, the protocol is guaranteed to converge asymptotically to the average consensus. After every linear iteration, each node must send its new value to its neighbors so that they can use it in the next iteration. For an extensive literature on average consensus protocol and its applications, check the surveys [5, 6] and the references therein.

To speed up the convergence, some approaches have focused on selecting the weights so that the convergence becomes faster. Xiao and Boyd in [7] have formulated the problem as a Semi Definite Program that can be efficiently and **globally** solved. However, speeding up the convergence does not reduce the number of messages that are sent in the network even when using the optimal weights. The reason is that the convergence is reached only asymptotically, and even if nodes' estimates are very close to the average, nodes keep on performing the averaging and sending messages to their neighbors.

The report is organized as follows: Section 2 gives the notation used and a formulation of the problem. Section 3 gives the previous work on the termination of the average consensus protocol. Section 4 motivates the work by an impossibility result for finite time termination. Section 5 gives the proposed algorithm, its analysis, and the simulations of the algorithm. Section 6 concludes the report.

2 Problem Formulation

Consider a network of n nodes that can exchange messages between each other through communication links. Every node in this network has a certain measurement (e.g. pressure or temperature), and we need each node to know the average of the initial measurements by following a distributed linear iteration approach. The network of nodes can be modelled as a graph $G = (V, E)$ where V is the set of vertices ($|V| = n$) and E is the set of edges such the $\{i, j\} \in E$ if nodes i and j are connected and can communicate (they are neighbors). Let also N_i be the neighborhood set of node i. Let $x_i(0) \in \mathbb{R}$ be the initial value at node i. We are interested in computing the average $x_{ave} = (1/n) \sum_{i=1}^{n} x_i(0)$, in a decentralized manner with nodes only communicating with their neighbors. Our network model will be the averaging done on a fixed network with a synchronization clock. When the clock ticks, all nodes in the system perform the iteration of the averaging protocol at the same time (this is the synchronous update assumption). At iteration $k + 1$, node i updates its state value x_i:

$$x_i(k+1) = w_{ii} x_i(k) + \sum_{j \in N_i} w_{ij} x_j(k) \tag{1}$$

where w_{ij} is the weight selected by node i for the value sent by its neighbor j and w_{ii} is the weight selected by node i for it own value. The matrix form equation is:

$$\mathbf{x}(k+1) = W\mathbf{x}(k) \tag{2}$$

where $\mathbf{x}(k)$ is the state vector of the system and W is the weight matrix.

Under some conditions on the weight matrix W given in [8], the system is guaranteed to converge to the average. In this report, we consider W to be a doubly stochastic matrix that satisfies these conditions and is constructed locally, some methods for constructing the weight W using only local information can be found is [9]. Let $\mathbf{1}$ be the vector having the value 1 in every entry, the convergence of the average consensus given above is in general asymptotic:

$$\lim_{k\to\infty} \mathbf{x}(k) = x_{ave}\mathbf{1}. \tag{3}$$

The averaging works as follows: after selecting the weights for their neighbors, a linear iteration phase starts. Each node calculates its new value depending on the weights and the neighbors' values, and then it broadcasts the new value to all its neighbors before the global clock ticks for the next iteration. Since the limit in (3) is typically reached at infinity, the nodes will be always busy sending messages. Let $M(k)$ be the number of nodes transmitting at iteration k, so without a termination protocol all nodes are transmitting at iteration k, $M(k) = M = n$, independent from the convergence of estimates. So how can nodes know that they are close enough to the average and thus stop sending messages to their neighbors and save bandwidth and energy? We propose in this report an algorithm that reduces communication overhead caused by sending messages at every iteration.

3 Related Work

Some work considers protocols for average consensus protocol to terminate in finite time. One approach is that estimates of nodes performing such a protocol reaches the exact average, as in [10], their solution is based on the concept of the *minimal polynomial* of the matrix W, where they showed that a node using coefficients of this polynomial can use its estimate over K successive iterations to calculate the average, but nodes must also have high memory capabilities to store the $n \times n$ matrix, and high processing capabilities to solve the n linearly independent equations to find the coefficients of the minimal polynomial. Another approach for finite time termination, is given in [11], but instead of exact average, guaranteed to be within a predefined threshold from the average. This approach runs three consensus protocols at the same time: the maximum and the minimum consensus restarted every U iterations where U is an upper bound of the diameter of the network, and average consensus. The difference between the maximum and minimum consensus provides a stopping criteria for nodes.

In a different time setting for average consensus, using the asynchronous randomized gossiping, the authors in [12] proposed an algorithm that leads to the termination of average consensus in finite time with high probability. In their approach, each node has a counter c_i that counts how many times the difference between the new estimate and the old one was less than a certain threshold τ, when the counter reaches a certain value, say C, the node will stop initiating the algorithm. They proved that by a correct choice of C and τ (depending on some parameters as the maximum degree in the network, the number of nodes, and the number of edges) the termination is with high probability.

However, a major drawback of these algorithms (without taking into consideration the memory capabilities they assumed nodes to have or the robustness of the system) the assumption of knowledge of some global variables at each nodes which violates the decentralized scenario of average consensus. Designing a decentralized algorithm for average consensus that terminates in finite time without using any global estimate (as the diameter of the network or the number of nodes) in the algorithm is still an open problem.

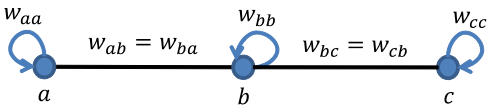


Figure 1: Line graph G with 3 nodes.

4 Motivation

We address the problem of termination of average consensus in this report. We will start by an impossibility result for termination of the average consensus protocol in finite time without using of a global estimate (mainly an upper bound on the diameter) and by only using the history estimates of the node.

Theorem 1. *In the average consensus protocol on a fixed graph (no link failure), where nodes perform synchronous iterations as in* (1) *to converge to the average of initial estimates, it is impossible to create a deterministic distributed algorithm that terminates in finite time to give the average of initial estimates or any bounded approximate of the average using only the history of the node's estimate.*

Proof. Consider a line graph G of three nodes, a,b, and c as in Fig. 1, where the weight matrix is real doubly stochastic matrix that satisfies the convergence conditions (as a result we can assume $w_{aa}, w_{cc} > 0$). Suppose there exists a termination algorithm for average consensus that terminates in finite time, then applying this algorithm on this graph implies that their exists an iteration $K > 0$ and $\epsilon > 0$ where node a decides to terminate and this algorithm uses only the history of estimates of node a: $x_a(0), x_a(1), x_a(2), ..., x_a(K)$, to decide to stop and $|x_a(K) - x_{ave}| < \epsilon$, where $x_{ave} = \alpha = \frac{x_a(0)+x_b(0)+x_c(0)}{3}$. We will define the extended mirror graph of G to be a line graph of 6 nodes a_1, a_2, b_1, b_2, c_1, and c_2, formed by two identical graphs as G but adding a $\{c_1, c_2\}$ edge. The initial estimates are the same for all nodes (e.g. for node a we have $x_{a_1}(0) = x_{a_2}(0) = x_a(0)$), weight matrix is also the same except for c_1 and c_2, where $w_{c_1c_1} = w_{c_2c_2} = w_{c_1c_2} = \frac{w_{cc}}{2}$ (see Fig. 2). Notice that on the new generated graph and the old one, $x_{a_1}(k) = x_a(k)\ \forall k \leq K$, so node a_1 on the new graph will decide to terminate after exactly K iterations. Notice now that we can continue doing this procedure of graph mirror extension till we have a line graph where $n > K$, call this graph H. a_1 in graph H still have the same estimates during the iterations $k \leq K$, so it will always terminate after K iterations. However, if we add a new node to H at the end of the line, having an estimate $\beta >> n\alpha$, the effect would only reach a_1 after a number of iterations $k > K$. Therefore, on this graph (the graph where we added one node at the end of the line graph H), the node a_1 will terminate with $|x_{a_1}(K) - \alpha| < \epsilon$, but it is not possible to guarantee it is close to the average because the new average now is $x_{ave} = \alpha + \frac{\beta-\alpha}{n} >> x_{a_1}(K)$. □

The above proof can be extended to include any graph G, not just line graphs, by using the same technique of generating the extended mirrors graphs of G to show that we cannot simply apply the termination protocol on any kind of networks not just the line graph.

The term termination in this decentralized scenario must refer to the messages sent by the nodes in the network but not to the execution of the algorithm at the level of nodes. The algorithms to use must allow nodes to refrain from sending their estimate as messages in some iterations and send it in another. The termination occurs when the number of messages sent in the network disappears, even if the nodes are still running the algorithm internally, but no

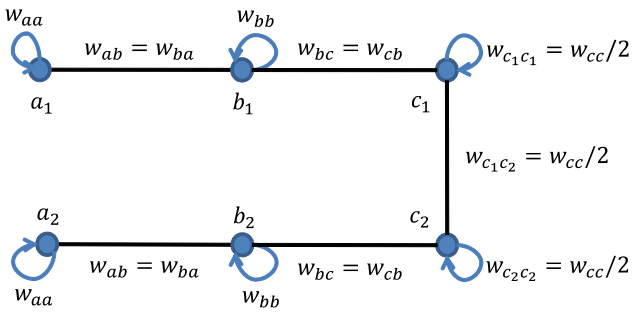


Figure 2: Extended mirror graph of G with 6 nodes.

more messages are sent in the network. In a less restrictive objective, we can also define the asymptotic termination when the rate of sending messages in the network is vanishing, i.e.:

$$\lim_{t\to\infty} \frac{\sum_{k=1}^{t} M(k)}{t} = 0. \tag{4}$$

In other words, the rate of messages in the network must give us an idea about the convergence of nodes, so if we monitor the network for an interval of iterations and saw that not many messages are generated, we should be able to conclude that the nodes are almost converged to the true average or to an approximate which is bounded by a threshold from the true average.

5 Our Approach

5.1 Centralized Termination

5.1.1 Introduction

Even if the nodes will not terminate in finite time, we are interested in decreasing the rate of the messages sent in the network in such a way to be related to the amount of improvement we are gaining in terms of the consensus. For example, if the nodes have wide range of difference in their estimates, sending messages can be efficient to decrease the error in the network and reach consensus. However, when the nodes are almost converged, sending a lot of messages will not be efficient, as the improvement is just a user perspective. So from an engineering perspective, our network must send more messages when the nodes have large differences in their estimates, and less messages when the system is almost converging. This subsection on centralized termination just provides basis and intuition for the more practical decentralized termination algorithm proposed in this report.

5.1.2 Overview

In this section, we discuss a simple centralized algorithm for termination of average consensus protocols. We call it a centralized protocol since in this protocol, each node can send a broadcast signal to the network to perform an iteration. At any iteration, if any of the nodes in the network sent this signal, all the nodes will respond by sending the new estimates to their neighbors according to the averaging equation:

$$\mathbf{x}(t+1) = W\mathbf{x}(t), \tag{5}$$

where t is the time when a broadcast signal is sent in the network. On the contrary, we see that if no signal is sent, the nodes will preserve the same estimate:

$$\mathbf{x}(t+1) = \mathbf{x}(t), \tag{6}$$

where t is the iteration where no broadcast message was sent. In the following, we will give a simple algorithm where the frequency of sending a broadcast signal will be decreasing (converging to zero i.e. no node in a network will be sending such a message), so the messages for averaging will also be sent less often as nodes converges to the average. This algorithm is an intuition to how the decentralized termination protocol works.

5.1.3 Analysis and Convergence

Let us define formally the method. Let $e(t)$ be an increasing scalar visible by all the nodes in the network such that $e(0) = 0$ and $0 \leq e(t) < \epsilon(t)$, where $\epsilon(t)$ is a boundary threshold which is also an increasing scalar such that $\epsilon(0) = \epsilon = constant$. Let W be the weight matrix of the network satisfying convergence conditions of average consensus and $\mathbf{x}(t)$ be the state vector of the system at iteration t. We define a logical expression L_t as a Boolean variable (either true or false) defined at every iteration t of the following expression:

$$L_t : \ e(t-1) + ||W\mathbf{x}(t-1) - \mathbf{x}(t-1)||_\infty < \epsilon(t-1), \tag{7}$$

with $L_0 = False$. According to this condition, actions are taken in iteration t. If this condition was false, a broadcast signal is sent in the network and all nodes will perform an iteration; we have $\mathbf{x}(t-1)$ changes, $\epsilon(t-1)$ also is increased by a value of ϵ/m^2, where m is a counter indicating how many times the value of ϵ has incremented in the past, but the other scalar $e(t-1)$ is kept the same. On the contrary, if L_t is true, then there is no signal in the network, and the nodes keep the same estimate as the previous iteration, the boundary threshold is also kept the same, but $e(t-1)$ is increased by $y(t-1) = ||W\mathbf{x}(t-1) - \mathbf{x}(t-1)||_\infty$. In particular, the changes to the network variables due to L_t are given in the following equation:

$$e(t) = \begin{cases} e(t-1) + y(t-1) & \text{if } L_t = True, \\ e(t-1) & \text{if } L_t = False, \end{cases} \tag{8}$$

$$\mathbf{x}(t) = \begin{cases} \mathbf{x}(t-1) & \text{if } L_t = True, \\ W\mathbf{x}(t-1) & \text{if } L_t = False, \end{cases} \tag{9}$$

$$\epsilon(t) = \begin{cases} \epsilon(t-1) & \text{if } L_t = True, \\ \epsilon(t-1) + \epsilon/k^2 & \text{if } L_t = False, \end{cases} \tag{10}$$

where k is a counter indicating how many times the value of $\epsilon(t)$ has incremented in the past.

When L_t is true, we call t a silent iteration because the nodes have the same estimate as the previous iteration (i.e. $x_i(t) = x_i(t-1)$) and there is no need to exchange messages of these estimates in the network. On the other hand, when L_t is false, we call t as a busy iteration because nodes will perform an averaging (i.e. $\mathbf{x}(t) = W\mathbf{x}(t-1)$) and the estimates must be exchanged in the network. Let us define T as the set of busy period:

$$T = \{t \mid L_t = False\}. \tag{11}$$

Let t_k where $k = 1, 2, ...$ be the increasing sequence of the elements of the set T. Let α_k be the number of silent iterations between t_k and t_{k+1}. We can see that $t_{k+1} = \alpha_k + t_k + 1$. We say that the algorithm is terminating asymptotically if the silent period is diverging, i.e. we have that:

$$\lim_{t\to\infty} \alpha_k = \infty. \tag{12}$$

First, it is not difficult to see that if $t \to \infty$, then $k \to \infty$ too, because an upper bound on k implies an upper bound on t. Therefore, it is sufficient to prove that $\lim_{k\to\infty} \alpha_k = \infty$.

Let $\mathbf{z}(k) = W\mathbf{x}(t_k) - \mathbf{x}(t_k)$, we can see that according to this algorithm,

$$\begin{aligned}\alpha_k &= \lfloor \frac{\epsilon(t_k) - e(t_k)}{||\mathbf{z}(k)||_\infty} \rfloor \\ &\geq \frac{\epsilon(t_k - 1) + \epsilon/k^2 - e(t_k - 1)}{||\mathbf{z}(k)||_\infty} - 1 \\ &\geq \frac{\epsilon}{k^2||\mathbf{z}(k)||_2} - 1. \end{aligned} \tag{13}$$

Moreover, $\mathbf{z}(k)$ evolve according to the following equation:

$$\mathbf{z}(k) = (W - G)\mathbf{z}(k - 1) = (W - G)^k\mathbf{z}(0), \tag{14}$$

where $G = 1/n\mathbf{1}\mathbf{1}^T$. Finally,

$$||\mathbf{z}(k)||_2 \leq C\rho^k(W - G), \tag{15}$$

where $C = ||\mathbf{z}(0)||$ and $\rho(W - G)$ is the spectral radius of the matrix $W - G$, so $0 < \rho < 1$ since W satisfies the condition of a converging matrix (see [8]). Putting everything together, we get finally that:

$$\alpha_k \geq \frac{\epsilon}{Ck^2\rho^k} - 1, \tag{16}$$

and hence $\alpha_k \to \infty$ as $k \to \infty$. Consequently, the rate of messages sent in the network is vanishing, namely

$$\lim_{t\to\infty} \frac{\sum_{i=1}^{t} M(i)}{t} = \lim_{t\to\infty} \frac{Mk}{t} = \lim_{t\to\infty} \frac{Mk}{k + \sum_{i=1}^{k-1} \alpha_i} = 0. \tag{17}$$

5.2 Decentralized Environment

5.2.1 Introduction

In a decentralized algorithm, each node works independently, the nodes only agree on synchronous time steps to do the iteration. The silent period on a node is just local, so a node can be silent, while its neighbor is not. In this scenario, we see that within an iteration, some nodes will be transmitting and others will be silent. This can cause instability in the network because the average with every iteration is not now conserved, and the scalar $e(k)$ defined in the previous subsection is now a vector $\mathbf{e}(k)$ where $e_i(k)$ is local to every node. To conserve the average in the decentralized setting, this scalar must take part in the state equation as we will show in what follows.

5.2.2 System Equation

In our approach, we consider a more general framework for average consensus where we study the convergence of the following equation:

$$\mathbf{x}(k+1) + \mathbf{e}(k+1) = W\mathbf{x}(k) + \mathbf{e}(k). \tag{18}$$

Some work have studied, the following equation as a perturbed average consensus and considered $\mathbf{e}(k)$ to be zero mean noise with vanishing variance (see [13, 14]). However, in our model, we consider $\mathbf{e}$ as a deterministic part of the state of the system and not a random variable. We consider sufficient conditions for the system to converge, and we use these conditions to give an algorithm that can reduce the number of messages sent in the network.

In the standard consensus algorithms, the state of the system is defined by the state vector $\mathbf{x}$, but in the modified system, the state equation is defined by the couple $\{\mathbf{x}, \mathbf{e}\}$.

Theorem 2. *Let $\mathbf{e}(k+1) = \mathbf{e}(k) - F(k)\mathbf{x}(k)$, and $A(k) = W + F(k)$. Assume the following conditions on the matrix $A(k)$:*

(a) $a_{ij}(k) \geq 0$ for all i, j, and k, and $\sum_{j=1}^{n} a_{ij}(k) = 1$ for all i and k.

(b) Lower bound on positive coefficients: there exists some $\alpha > 0$ such that if $a_{ij}(k) > 0$, then $a_{ij}(k) \geq \alpha$, for all i, j, and k.

(c) Positive diagonal coefficients: $a_{ii}(k) \geq \alpha$, for all i, k.

(d) Cut-balance: for any i with $a_{ij}(k) > 0$, we have j with $a_{ji}(k) > 0$.

(e) $\lim_{k\to\infty} \mathbf{x}(k) = \mathbf{x}^\star \Rightarrow \lim_{k\to\infty} F(k)\mathbf{x}(k) = \mathbf{0}$.

Then $\lim_{k\to\infty} \mathbf{x}(k) = x'_{ave}\mathbf{1}$, where we have that $x'_{ave} \in [\min_j x_j(0), \max_j x_j(0)]$ if furthermore we have $\mathbf{e}(0) = \mathbf{0}$ and $e_i(k) < \epsilon$, then $|x_{ave} - x'_{ave}| < \epsilon$.

Proof. Let us first prove that $\mathbf{x}(k)$ converges. By substituting the equation of $\mathbf{e}(k+1)$ in (18), we obtain:

$$\mathbf{x}(k+1) = A(k)\mathbf{x}(k), \tag{19}$$

where $A(k) = W + F(k)$. From the conditions (a),(b),(c), and (d) on $A(k)$, we have from [15] that $\mathbf{x}$ converges, i.e. $\lim_{k\to\infty} \mathbf{x}(k) = \mathbf{x}^\star$. Since the system is converging, then from equation (18), we can see that:

$$\mathbf{x}^\star + \lim_{k\to\infty} (\mathbf{e}(k+1) - \mathbf{e}(k)) = W\mathbf{x}^\star,$$

so,

$$\mathbf{x}^\star = W\mathbf{x}^\star. \tag{20}$$

Therefore, $\mathbf{x}^\star$ is an eigenvector corresponding to the highest eigenvalue ($\lambda_1 = 1$) of W. So we can conclude that $\mathbf{x}^\star = x'_{ave}\mathbf{1}$ where x'_{ave} is a scalar (Perron-Frobenius theorem).

The condition $\mathbf{1}^T W = \mathbf{1}^T$ on the matrix W in equation (2) leads to the preservation of the average in the network, $\mathbf{1}^T\mathbf{x}(k) = nx_{ave}\ \forall k$. This condition is not necessary satisfied by $A(k)$, so let us prove now that the system preserves the average x_{ave}:

$$\mathbf{1}^\top(\mathbf{x}(k+1) + \mathbf{e}(k+1)) = \mathbf{1}^\top(W\mathbf{x}(k) + \mathbf{e}(k)) \tag{21}$$

$$= \mathbf{1}^\top(\mathbf{x}(k) + \mathbf{e}(k)). \tag{22}$$

The last equality comes from the fact that W is sum preserving since $\mathbf{1}^\top W = \mathbf{1}^\top$.

Finally by a simple recursion we have that $\mathbf{1}^\top(\mathbf{x}(k) + \mathbf{e}(k)) = \mathbf{1}^\top\mathbf{x}(0) = nx_{ave}$, and the average is conserved. Moreover, since $|e_i(k)| \leq \epsilon \ \forall k$, so we have:

$$|(1/n)\mathbf{1}^\top\mathbf{x}(k) - x_{ave}| \leq \epsilon \ \forall k. \tag{23}$$

But we just proved that $\lim_{k\to\infty} \mathbf{x}(k) = x'_{ave}\mathbf{1}$, so this consensus is within ϵ from the desired x_{ave}:

$$|x'_{ave} - x_{ave}| \leq \epsilon. \tag{24}$$

This ends the proof. □

5.3 Message Reducing Algorithm

We try to solve the termination problem through a *fully decentralized* approach. We consider large-scale networks where nodes have limited resources (in terms of power, processing, and memory), do not use any global estimate (e.g. diameter of the network or number of nodes), keep only one iteration history, and can only communicate with their neighbors. Our main goal is that the number of messages in the network is affected by how close the nodes are to consensus. Namely, the closer they are to consensus, the less nodes transmit messages at each iteration. The algorithm must guarantee that the system reaches a consensus within a certain predefined margin from the average.

The main idea is that a node, say i for example, will compare its new calculated value with the old one. According to the change in the estimate, i will decide either to broadcast its new value or not to do so. We divide an iteration into two parts, in the first part of the iteration, only nodes with significant change in their estimates are allowed to send messages. However, in the second part of the iteration, only nodes polled by their neighbors from phase 1 are allowed to send an update. As a node approaches the consensus, the change in its estimate becomes smaller, and more nodes will stop sending messages during an iteration of the algorithm, and the messages in the system will disappear asymptotically. During the silent period of each node, it accumulates an error caused by the absence of its messages. When the silent node transmits again, it can always reduce the accumulated error by redistributing it in the network and thus saving messages while preserving the average.

Before starting the linear iterative equation, nodes will select weights as in the standard consensus algorithm. The weight matrix considered here must be doubly stochastic with $0 < \alpha < w_{ii} < 1 - \alpha < 1$ for some constant α. Each node i in the network keeps two state values at iteration k:

- $x_i(k)$: the estimate of node i used in the iterative equations by the other nodes.
- $e_i(k)$: a real value that monitors the shift from the average due to the iterations where node i did not send a message to its neighbors. It is initially set to zero, $e_i(0) = 0$.

Each node also keeps its own boundary threshold $\epsilon_i(k)$ where $\epsilon_i(1) = \frac{\epsilon}{2} = constant \ \forall i$. Note that this epsilon is increased after every transmission as in the centralized case, but the difference here is that it is local to every node.

Each iteration is divided into two phases:
In the first phase, a node i can be in one of the two following states:

- *Transmit*: The set of nodes corresponding to this state is T_k, where the subindex k corresponds to the fact that the set can change with every iteration k. The nodes in T_k send their new calculated estimate to their neighbors. They also poll the nodes having maximum and minimum estimates in their neighborhood to transmit in phase 2.

Algorithm 1 Termination Algorithm -node i- Phase 1

$\{x_i(k), e_i(k)\}$ are the state values of node i at iteration k, $0 < \alpha < w_{ii} < 1 - \alpha < 1$, $counter_i = 1$ is the counter for the number of transmissions so far. $\epsilon_i(1) = \epsilon/2 \in \mathbb{R}$, T_k is set of $Transmit$ state. W_k set corresponding to $Wait$ state. Initially we have $T_k = W_k = \varnothing$. Every node i follows the following algorithm at iteration k.
$y_i(k+1) \leftarrow w_{ii}x_i(k) + \sum_{j \in N_i} w_{ij}x_j(k)$
$d_i \leftarrow y_i(k+1) - x_i(k) + e_i(k)$
if $|d_i| < \epsilon_i(counter_i)$ **then**
  i changes to a *Wait* state. $\setminus\setminus\, i \in W_k$
else
  $counter_i = counter_i + 1$
  $\epsilon_i(counter_i) = \epsilon_i(counter_i - 1) + \epsilon_i(1)/counter_i^2$
  $c_i \leftarrow \frac{\alpha}{(1-w_{ii})}\Big(y_i(k+1) - x_i(k)\Big)$
  if $|c_i| \leq |e_i(k)|$ **then**
    $x_i(k+1) \leftarrow y_i(k+1) + \text{sign}(c_i.e_i(k))c_i$
    $e_i(k+1) \leftarrow e_i(k) - \text{sign}(c_i.e_i(k))c_i$
  else
    $x_i(k+1) \leftarrow y_i(k+1) + e_i(k)$
    $e_i(k+1) \leftarrow 0$
  end if
  i changes to a *Transmit* state. $\setminus\setminus\, i \in T_k$
  Notify the neighbors having maximum and minimum values.
end if
Go to Phase 2

- *Wait*: The set of nodes corresponding to this state is W_k. The node's decision will be taken in the second phase of the iteration based on the action of nodes in the *Transmit* state (depending if they were polled by any of their neighbors).

In the second part of the iteration, nodes that are in W_k will be classified as follows:

- *Silent*: The set of nodes corresponding to this state is S_k. These are the nodes that will remain silent with no message sent from their part in the network. The nodes in S_k have that non of their neighbors sending them any poll message.

- *Cut-Balance*: The set of nodes corresponding to this state is B_k. They are called $Cut - Balance$ because they insure the cut-balance condition (d) of theorem 2. They are the nodes in W_k that have been polled by at least one neighbor in T_k .

The two phases of the termination protocol implemented at each node are described by pseudocode in Algorithm 1 and 2. Nodes in the T_k set (the set of nodes that are in a $Transmit$ state) will broadcast their estimate to their neighbors at the end of the first phase, while nodes in W_k set (or $Wait$ state) will postpone their decision to send or not till the next phase. Nodes that do not receive a message from their neighbors at a certain iteration, uses the last seen estimate from the specified neighbors (note: absence of messages from a neighbor during an iteration does *not* mean the failure of link, it means that the neighbor is broadcasting the same old estimate as before, so we may differentiate the link failure by a "keep alive" message sent frequently to maintain connectivity and set of neighbors). The **input** for the algorithm are the estimates of the neighbor of i, the weights selected for these neighbors, and the state values $\{x_i(k), e_i(k)\}$.

Algorithm 2 Termination Algorithm - Phase 2

1: $\{x_i(k), e_i(k)\}$ are the state values of node i at iteration k.
2: **for all** nodes i having *Wait* state **do**
3: $y_i(k+1) \leftarrow w_{ii}x_i(k) + \sum_{j\in N_i} w_{ij}x_j(k)$
4: **if** i received a poll message from any neighbor **then**
5: $z_i(k+1) \leftarrow (w_{ii} + \sum_{j\in N_i\cap W_k} w_{ij})x_i(k) + \sum_{j\in N_i\cap T_k} w_{ij}x_j(k)$
6: $x_i(k+1) \leftarrow z_i(k+1)$
7: $e_i(k+1) \leftarrow y_i(k+1) - z_i(k+1) + e_i(k)$
8: i changes to a $Cut-Balance$ state. $\setminus\setminus\, i \in B_k$
9: **else**
10: $x_i(k+1) \leftarrow x_i(k)$
11: $e_i(k+1) \leftarrow y_i(k+1) - x_i(k) + e_i(k)$
12: i changes to a *Silent* state. $\setminus\setminus\, i \in S_k$
13: **end if**
14: **end for**
15: $k+1 \leftarrow k$

The **output** of the first phase is the new state values $\{x_i(k+1), e_i(k+1)\}$ for nodes in T_k and the output of the second phase is the new state values $\{x_i(k+1), e_i(k+1)\}$ for nodes in W_k. Let us go through the lines of the algorithm. In phase 1, $y_i(k+1)$ of line 2 is the weighted average of the estimates received by node i; without the termination protocol this value would be sent to all its neighbors. The protocol evaluates how much $y_i(k+1)$ differs from the state value $x_i(k)$. This difference accumulates in d_i in line 3. If this shift is less than a given threshold ϵ_i, the node will wait for next phase to take decision. If the condition in line 4 is not satisfied, that means the node will send a new value to its neighbors. Lines $7-8$ concerns the extending of the boundary threshold $\epsilon_i(k)$ after every transmission. Note that by this extension method, we have $\epsilon_i(k) < \epsilon\ \forall i, k$ since

$$\begin{aligned}\lim_{k\to\infty} \epsilon_i(k) &= \epsilon_i(1)(\sum_{i=1}^{\infty} 1/k^2) \\ &< \epsilon_i(1) \times 2 \\ &= \epsilon. \end{aligned} \tag{25}$$

We introduce in line 9 a new scalar c_i used for deciding which portion of $e_i(k)$ the node will send in the network. In lines $11-12$ and $14-15$, the algorithm satisfies the equation (18). Then the new state value $x_i(k+1)$ is sent to the neighbors and $e_i(k+1)$ is updated accordingly. In Phase 2 of the algorithm (Algorithm 2), nodes initially in the wait state will decide either to send a cut-balance massage or to remain silent, the cut balance messages are sent when a node receives a poll message from any of its neighbors.

5.4 Convergence study

The convergence of the previous algorithm is mainly due to the fact that the proposed algorithm satisfies the conditions of convergence given in 5.2.2. In fact, the algorithm is designed to satisfy all these conditions that guarantee convergence. Starting with the state equation, we can notice from the algorithm 1 given that whatever the condition the nodes face, it is always true that the sum of the new generated state values $\{x_i(k+1), e_i(k+1)\}$ is as follows:

$$x_i(k+1) + e_i(k+1) = y_i(k+1) + e_i(k),$$

where $y_i(k+1) = w_{ii}x_i(k) + \sum_{j\in N_i} w_{ij}x_j(k)$. As a result the system equation is the one studied in section 5.2.2 (equation (18)).

Before going through the different conditions in the theorem 2, we should also show that $\mathbf{e}(k+1) = \mathbf{e}(k) - F(k)\mathbf{x}(k)$ for some matrix $F(k)$ such that $F(k)\mathbf{1} = \mathbf{0}$. According to the algorithm we can write,

$$e_i(k+1) = e_i(k) - v_i(k), \tag{26}$$

where $v_i(k)$ differs according to the state of the node i, but it only depends on the estimate of node i and its neighbors:

$$v_i(k) = \begin{cases} \pm\frac{\alpha}{(1-w_{ii})}\Big(y_i(k+1) - x_i(k)\Big) & \text{if } i \in T_k - (1), \\ \pm\frac{\alpha\gamma}{(1-w_{ii})}\Big(y_i(k+1) - x_i(k)\Big) & \text{if } i \in T_k - (2), \\ x_i(k) - y_i(k+1) & \text{if } i \in S_k, \\ z_i(k+1) - y_i(k+1) & \text{if } i \in B_k, \end{cases} \tag{27}$$

where $T_k - (1)$ is the set of nodes subset in T_k where $|c_i| \leq |e_i(k)|$, and $T_k - (2)$ set of nodes where $|c_i| > |e_i(k)|$. In the latter case, $e_i(k+1) = 0$, but we can always find $\gamma < 1$ such that $e_i(k+1) = e_i(k) - \gamma(\text{sign}(c_i.e_i(k)c_i) = 0$ where $c_i = \frac{\alpha}{(1-w_{ii})}(y_i(k+1) - x_i(k))$. $y_i(k+1)$ and $z_i(k+1)$ are as indicated in the algorithm and are a linear combination of the elements of $\mathbf{x}(k)$. From the equation of $v_i(k)$, we can also see that it is a linear combination of the elements of $\mathbf{x}(k)$, such that the coefficients sum to 0. A row i in $F(k)$ will be the coefficients of the estimates $\mathbf{x}(k)$ in $v_i(k)$, so $F(k)\mathbf{1} = \mathbf{0}$.

Now we can study the conditions mentioned in the theorem 2 on the matrix $A(k) = W + F(k)$.

Lemma 1. *$A(k)$ is a stochastic matrix that satisfies conditions (a),(b),and (c) of Theorem 2.*

Proof. First, we can see that $A(k)\mathbf{1} = \mathbf{1}$ since $W\mathbf{1} = \mathbf{1}$ and $F(k)\mathbf{1} = \mathbf{0}$. It remains to prove that all entries in the matrix $A(k)$ are non negative.

We will prove this by considering each row i of $A(k)$ according to the action taken by node i. We can distinguish four cases:

1. Node $i \in T_k$ - condition 1: $|c_i| \leq |e_i(k)|$
   $a_{ii} = w_{ii} - \frac{\alpha}{1-w_{ii}} \times (1 - w_{ii}) = w_{ii} - \alpha > 0$ since $w_{ii} > \alpha$
   and $a_{ij} = w_{ij} + \frac{\alpha}{1-w_{ii}} \times w_{ij} \geq w_{ij} > 0 \quad \forall j \in N_i$.
   or
   $a_{ii} = w_{ii} + \frac{\alpha}{1-w_{ii}} \times (1 - w_{ii}) > \alpha$ since $w_{ii} > \alpha$
   and $a_{ij} = w_{ij} - \frac{\alpha}{1-w_{ii}} \times w_{ij} > 0 \quad \forall j \in N_i$ since $\alpha < w_{ii} < 1 - \alpha$.

2. Node $i \in T_k$ - condition 2: $|c_i| > |e_i(k)|$
   since $|c_i| > |e_i(k)|$, we can always find positive $\gamma < 1$, such that
   $a_{ii} = w_{ii} - \frac{\gamma\alpha}{1-w_{ii}} \times (1 - w_{ii}) = w_{ii} - \gamma\alpha > 0$ since $w_{ii} > \alpha$
   and $a_{ij} = w_{ij} + \frac{\gamma\alpha}{1-w_{ii}} \times w_{ij} \geq w_{ij} > 0 \quad \forall j \in N_i$.
   or
   $a_{ii} = w_{ii} + \frac{\gamma\alpha}{(1-w_{ii})} \times (1 - w_{ii}) > \alpha$ since $w_{ii} > \alpha$
   and $a_{ij} = w_{ij} - \frac{\gamma\alpha}{1-w_{ii}} \times w_{ij} > (1 - \frac{\alpha}{\alpha})w_{ij} > 0 \quad \forall j \in N_i$.

3. Node $i \in S_k$:
   then $a_{ii} = w_{ii} + (1 - w_{ii}) = 1$
   and $a_{ij} = 0 \quad \forall j \neq i$.

4. Node $i \in B_k$:
   then $a_{ii} \geq w_{ii} > 0$
   and $a_{ij} \in \{w_{ij}, 0\} \quad \forall j \neq i$.

Therefore, $A(k)$ is stochastic at every iteration k.

□

Definition 1. *Two matrices, A and B, are said to be equivalent with respect to a vector* $\mathbf{v}$ *if and only if* $A\mathbf{v} = B\mathbf{v}$.

Notice that $A(k)$ satisfies conditions (a),(b), and (c) of Theorem 2, but possibly not the cut balance condition (d) because for a node $i \in T_k$ that transmits, $a_{ij}(k) > 0 \ \forall \ j \in N_i$, but it can be that $\exists j \in N_i$ such that $a_{ji} = 0$ if j was silent at that iteration ($j \in S_k$). However, the next theorem shows that there is a matrix $B(k)$ equivalent to $A(k)$ with respect to $\mathbf{x}(k)$ that satisfies all the conditions.

Lemma 2. *For all k, there exists a matrix* $B(k)$ *equivalent to* $A(k)$ *with respect to* $\mathbf{x}(k)$ *such that* $B(k)$ *satisfies the conditions (a),(b),(c), and (d) of Theorem* 2.

Proof. Let $m(k) = \text{argmin}_{j \in N_i} x_j(k)$ and $M(k) = \text{argmax}_{j \in N_i} x_j(k)$, we will proof the existence of $B(k)$ by modifying $A(k)$ in such a way to preserve the properties (a) to (c) and to add the new property (d). For simplicity of notation we will drop k from the variables since this is true for every k. Note first that the condition (d) is not satisfied in A only for the rows where i belongs to T_k, let $\mathbf{a}_i$ denote the row i of A and $\mathbf{b}_i$ denote the row i of B. Since any node i in T_k must poll the nodes m and M to transmit in Phase 2, then we are sure that the column i has at least three non zero elements (a_{ii}, a_{mi}, and a_{Mi}). Let $C_i = \{j \mid a_{ji} > 0 \ , \ i \neq j\}$, so we are sure that C_i contains at least two elements m and M. The cut balance condition requires that the row of i must only have positive values at the index where the column is positive. We can write that

$$\begin{aligned}
\mathbf{a}_i\mathbf{x}(k) &= a_{ii}x_i(k) + \sum_{j \in N_i} a_{ij}x_j(k) \\
&= a_{ii}x_i(k) + \sum_{j \in C_i} a_{ij}x_j(k) + \sum_{j \in N_i - C_i} a_{ij}x_j(k) \\
&= a_{ii}x_i(k) + \sum_{j \in C_i} a_{ij}x_j(k) + hx_M(k) + fx_m(k), \qquad (28)
\end{aligned}$$

where $h = \frac{\sum_{j \in N_i - C_i} a_{ij}(x_j - x_m)}{x_M - x_m}$, $f = \frac{\sum_{j \in N_i - C_i} a_{ij}(x_M - x_j)}{x_M - x_m}$. Since $h \geq 0$, $f \geq 0$ and $f + h = \sum_{j \in N_i - C_i} a_{ij}$, we can define the row vector $\mathbf{b}_i$ to be:

$$\mathbf{b}_i := \begin{cases} b_{ij} = a_{ij} & \text{if } j = i, \\ b_{ij} = a_{ij} + f & \text{if } j = m, \\ b_{ij} = a_{ij} + h & \text{if } j = M, \\ b_{ij} = a_{ij} & \text{if } j \in C_i - m - M, \\ b_{ij} = 0 & \text{if } j \in N_i - C_i. \end{cases} \qquad (29)$$

Finally, $\mathbf{b}_i\mathbf{x}(k) = \mathbf{a}_i\mathbf{x}(k)$ and it satisfies the conditions (a) to (c), so $\forall i \in T_k$, we replace $\mathbf{a}_i$ by $\mathbf{b}_i$ and we get the new matrix B which is equivalent to A with respect to $\mathbf{x}(k)$. □

Lemma 3. *The message reduction algorithm (Phases 1, 2) satisfies condition (e) of Theorem* 2.

Proof. We will prove it by contradiction. Suppose $\lim_{k\to\infty}\mathbf{x}(k)=\mathbf{x}^\star$, but $\lim_{k\to\infty}F(k)\mathbf{x}(k)\neq\mathbf{0}$, then there exists a node i such that $\lim_{k\to\infty}x_i(k)=x_i^\star$ and $\lim_{k\to\infty}y_i(k+1)=w_{ii}x_i^\star+\sum_{j\in N_i}w_{ij}x_j^\star=y_i^\star$, but $y_i^\star-x_i^\star=\delta^\star>0$. From Algorithm 1, we can see that the node will enter a transmit state infinitely often (because d_i increases linearly with δ^* and it will reach the threshold ϵ_i). Then, the node i will update its estimate according to the equation

$$x_i(k+1)=y_i(k+1)+\frac{\alpha}{1-w_{ii}}(y_i(k+1)-x_i(k)).$$

Letting $k\to\infty$ yields

$$(1+\frac{\alpha}{1-w_{ii}})\delta^\star=0.$$

Thus, $\delta^\star=0$ which is a contradiction, and the algorithm satisfies condition (e) of Theorem 2. □

The algorithm also provides that $|e_i(k)|\leq\epsilon_i(k)\ \forall k,i$ and $\epsilon_i(k)<\epsilon\ \forall i,k$, as in the first phase this condition is satisfied by construction, and for the second phase of the iteration, nodes from Phase 1 can check for worst case analysis and they only enter into *Wait* state if they are sure that the condition can be satisfied in the next phase iteration.

Now we are ready to state the main Theorem in this section:

Theorem 3. *The nodes applying the message reducing algorithm given in pseudocode by Algorithm 1 and 2, have estimates converging to a consensus within a margin ϵ from x_{ave}, i.e.* $\lim_{k\to\infty}\mathbf{x}(k)=x'_{ave}\mathbf{1}$ *and* $|x'_{ave}-x_{ave}|\leq\epsilon$.

Proof. The theorem is due to the fact that the Lemmas given in this subsection show that the algorithm satisfies all the convergence conditions of Theorem 2. □

5.5 Asymptotic Termination

We have proved in the previous section that the algorithm proposed is converging to a consensus. We will use this to show that any node i can increase the boundary threshold $\epsilon_i(k)$ at certain times $k_s,\ s=0,1,2,...$ (i.e. $\epsilon_i(k_s)=\epsilon_i(k_{s-1})+\epsilon(0)/s^2$) such that the silent period of this node ($\alpha_i(s)$, the consecutive number of iterations that a node i did not send any message) is growing to infinity. The system equation we had following our message saving algorithm is as follows:

$$\mathbf{x}(k+1)=A(k)\mathbf{x}(k), \tag{30}$$

where A(k) was a stochastic matrix.

Let us define for a matrix A the proper coefficient of ergodicity $\tau(A)$ to study the convergence rate of the system (for more information on proper coefficients of ergodicity see [16]),

$$\tau(A)=\frac{1}{2}\max_{i,j}\sum_{s=1}^{n}|a_{is}-a_{js}|. \tag{31}$$

Let also $U_{p,r}$ be the r-step backward product of the matrix A(k):

$$U_{p,r}=A(p+r)...A(p+2)A(p+1). \tag{32}$$

Theorem 4. *Ergodicity of backwards products $U_{p,r}$ formed from a given sequence $A(k),k\geq1$, obtains if and only if there is a strictly increasing sequence of positive integers $\{k_s\},\ s=0,1,2,...$ such that*

$$\sum_{s=0}^{\infty}\{1-\tau(U_{k_s,k_{s+1}-k_s})\}=\infty. \tag{33}$$

Proof. [16, p.155] . □

Since our system is ergodic (converges) as we proved in previous section, this theorem implies that for any positive constant $\delta < 1$ we have a strictly increasing sequence of positive integers $\{k_s\}$, $s = 0, 1, 2, ...$ such that

$$\tau(U_{k_s, k_{s+1}-k_s}) \leq \delta. \tag{34}$$

By the increment of the threshold $\epsilon_i(k)$ at instances k_s, we have that the silent period $\alpha_i(s)$ of a node i is the number of iterations node i is silent after its last increment at time k_s. The silent period satisfies the following equation:

$$\alpha_i(s) \geq \frac{\epsilon}{s^2 \big(W\mathbf{x}(k_s) - \mathbf{x}(k_s)\big)_i}, \tag{35}$$

let $\alpha(s)$ be the minimum across all local $\alpha_i(s)$, we have:

$$\alpha(s) \geq \frac{\epsilon}{s^2 ||W\mathbf{x}(k_s) - \mathbf{x}(k_s)||_\infty}. \tag{36}$$

Since W is a stochastic matrix, then all the values of the vector $W\mathbf{x}(k_s)$ belong to the interval $[\min_i x_i(k_s), \max_i x_i(k_s)]$, so we have,

$$||W\mathbf{x}(k_s) - \mathbf{x}(k_s)||_\infty \leq \max_{i,j} |x_i(k_s) - x_j(k_s)|, \tag{37}$$

but from the definition of the coefficient of ergodicity we have that

$$\max_{i,j} |x_i(k_s) - x_j(k_s)| \leq \tau(U_{k_{s-1}, k_s - k_{s-1}}) \times \{\max_{i,j} |x_i(k_{s-1}) - x_j(k_{s-1})|\}, \tag{38}$$

and we can conclude that the silent period is growing exponentially in s:

$$\alpha(s) \geq \frac{\epsilon}{C s^2 \delta^s}, \tag{39}$$

where C is just a constant depends on the initial state vector $\mathbf{x}(0)$.

And the asymptotic termination is obtained:

$$\lim_{s \to \infty} \alpha(s) = \infty. \tag{40}$$

Even though nodes are not aware of these time k_s where they have to increment the threshold, with an incrementation $\epsilon_i(m)$ at times m, the system is very robust is the sense that nodes do not have to be synchronized and errors in wrong time incrementation can also be tolerated and do not affect the convergence as long as $m^2 = o(\delta^s)$, where $o(.)$ is the small-oh notation (in the study of convergence m was equal to s). In the simulations and as shown in the algorithm 1, we incremented m ($counter_i$ in the algorithm) whenever a node is going to transmit at phase one, and the results are satisfactory.

5.6 Simulations

To simulate the asymptotic termination of the algorithm described above, we considered two types of graphs with $n = 50$, the Random Geometric Graphs (RGG) with connectivity radius $r = 0.234$, and the Erdos Renyi (ER) with average degree 4. All the graphs considered are

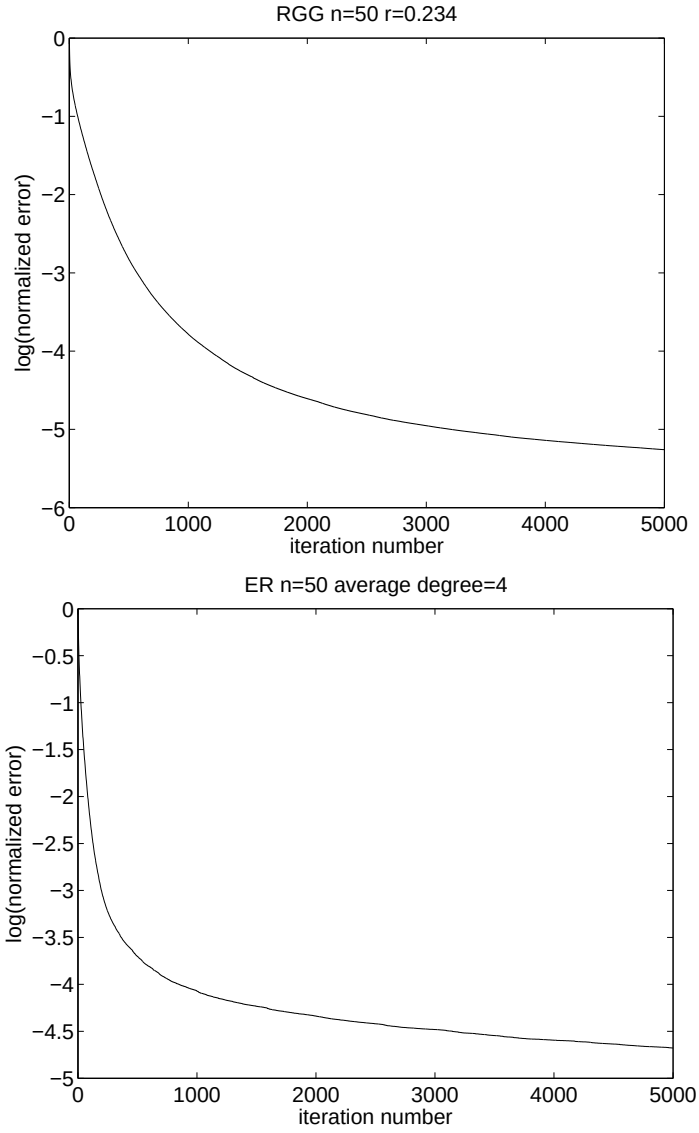


Figure 3: Normalized error on RGG and ER graphs.

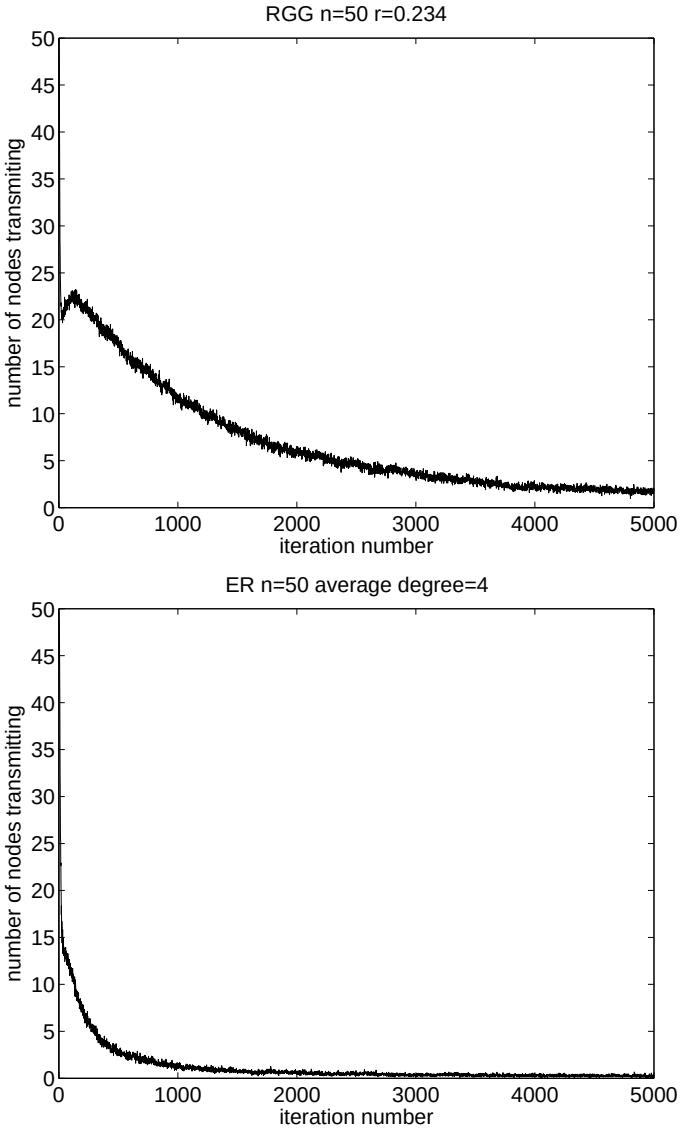


Figure 4: Messages sent with every iteration.

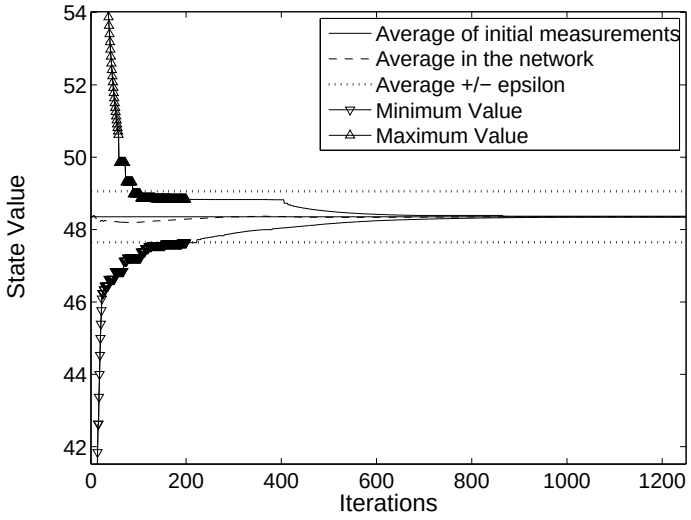


Figure 5: Convergence to a consensus.

connected. In Fig. 3, we show how the error is decreasing on the two types of the networks, while on Fig. 4, we show the number of nodes transmitting at a given iteration. Note that the curves are averaged across 100 independent runs to have a good confidence interval.

The simulations confirm what we have established theoretically. That is, the number of messages sent when the nodes are close to consensus is much less than when they start. To measure the distance from the average, we considered the normalized error metric defined as,

$$normalized\ error(k) = \frac{||\mathbf{x}(k) - \bar{\mathbf{x}}(k)||_2}{||\mathbf{x}(0) - \bar{\mathbf{x}}(0)||_2},$$

where $\bar{\mathbf{x}}(k) = \frac{\mathbf{1}^T\mathbf{x}(k)}{n}\mathbf{1}$. Note that the initial condition here was giving to each node a uniformly random value between 0 and 10. With only 1000 iteration on Erdos Renyi graphs, on average, less than one node is transmitting in the network. While it takes a bit more iterations for message to be reduced on RGG, this is due to the fact that the speed of convergence is effected by the structure of the graph and thus it affects the silent time of nodes. But we are sure from the above study that the rate of sending messages is disappearing asymptotically while nodes are reaching convergence x'_{ave} bounded by ϵ from x_{ave} as the next figure shows.

In Fig. 5 the maximum and minimum values in a random geometric graph which show the contraction toward a consensus x'_{ave} within ϵ from x_{ave}. The initial values are samples from i.i.d. uniform random variable with support $[0, 100]$. The figure also shows that x'_{ave} is very close to x_{ave}. In the termination algorithm we proposed, $x_{max}(k)$ and $x_{min}(k)$ are non-increasing and non-decreasing functions respectively. This is clear from the figure.

In a dynamic case, we assumed that at certain iterations some nodes for a certain reason change their estimates to a completely different one. In the real life, if nodes where measuring temperature for example, this change can be due to the change in the temperature itself. We considered an Erdos Renyi graph with 50 nodes and average degree 4, as an initial state, nodes estimate takes a value in the interval $[0, 10]$ uniformly at random. Every 1000 iterations, on average, 10% of the nodes restart there estimate by a new one in the interval $[10, 20]$ chosen

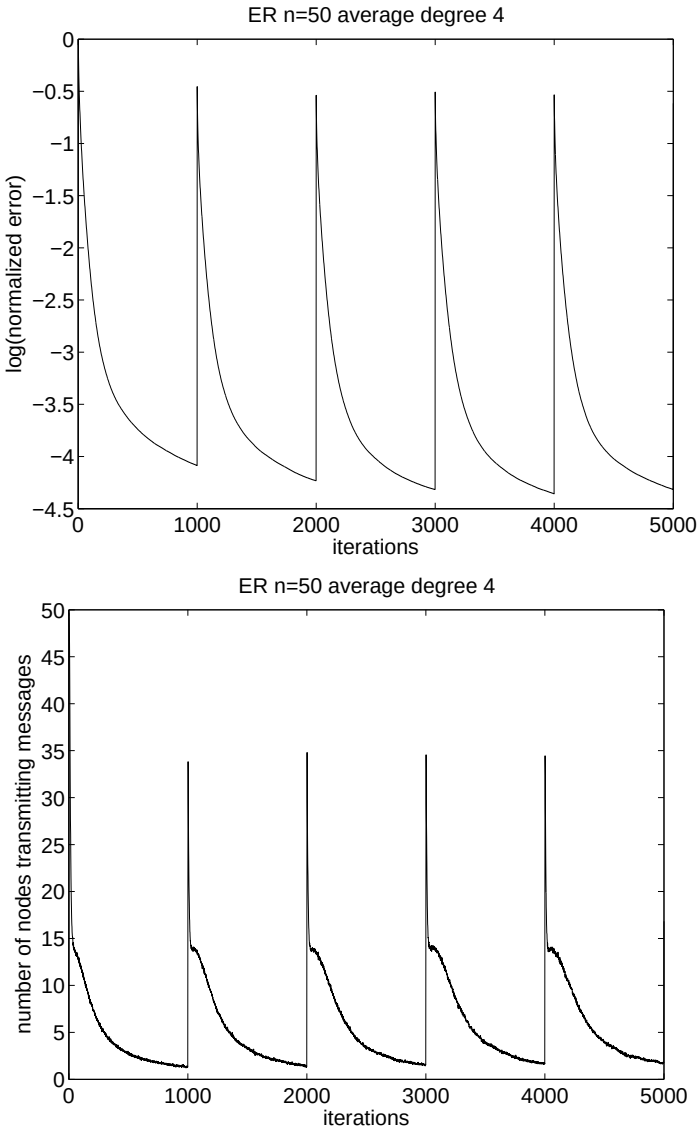


Figure 6: Normalized error and number of nodes transmitting on the dynamic ER graph, where every 1000 iterations 10% of the nodes changes their estimates.

uniformly at random. This change would lead of course to the instability of the protocol, but we show in the simulations, that the algorithm we use is able to cope with this change and the system is automatically adapting to this dynamic behavior. With every change in the estimates, the network give a burst of messages to stabilize the network to the new average. Fig. 6 shows how the normalized error is effected by the change of estimates and how the system following our algorithm increase the number of messages, so that the error decreases again and the network is stabilized. The simulations in this part are averaged across 1000 independent runs on connected ER graphs.

6 Conclusion

In this report, we gave an algorithm to terminate the messages sent in average consensus. The algorithm is totally decentralized and does not depend on any global variable, it only uses the weights selected to neighbors and one iteration history of the estimates to decide to send a message or not. We proved that this algorithm is converging to a consensus at most ϵ from the true average. The algorithm is also robust and adaptive to errors caused by a node suddenly changing its estimate to a different one.

References

[1] W. Ren and R. W. Beard, *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[2] N. Hayashi and T. Ushio, "Application of a consensus problem to fair multi-resource allocation in real-time systems." in *CDC*. IEEE, 2008, pp. 2450–2455.

[3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2508–2530, June 2006.

[4] F. Bénézit, A. G. Dimakis, P. Thiran, and M. Vetterli, "Order-optimal consensus through randomized path averaging," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5150 –5167, October 2010.

[5] R. Olfati-saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," in *Proc. of the IEEE*, Jan. 2007.

[6] W. Ren, R. Beard, and E. Atkins, "A survey of consensus problems in multi-agent coordination," in *American Control Conference, 2005. Proceedings of the 2005*, June 2005, pp. 1859 – 1864 vol. 3.

[7] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2004.

[8] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM REVIEW*, vol. 46, pp. 667–689, April 2004.

[9] K. Avrachenkov, M. El Chamie, and G. Neglia, "A local average consensus algorithm for wireless sensor networks," *LOCALGOS international workshop held in conjunction with IEEE DCOSS'11*, June 2011.

[10] S. Sundaram and C. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *American Control Conference (ACC '07)*, July 2007, pp. 711 –716.

[11] V. Yadav and M. V. Salapaka, "Distributed protocol for determining when averaging consensus is reached," *Forty-Fifth Annual Allerton Conference Allerton House, UIUC, Illinois, USA*, September 2007.

[12] A. Daher, M. Rabbat, and V. K. Lau, "Local silencing rules for randomized gossip," *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2011.

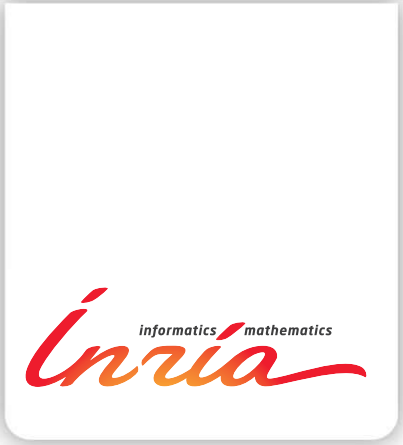
[13] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *J. Parallel Distrib. Comput.*, vol. 67, pp. 33–46, January 2007.

[14] Y. Hatano, A. Das, and M. Mesbahi, "Agreement in presence of noise: pseudogradients on random geometric networks," in *in Proc. of the 44th IEEE CDC-ECC*, December 2005.

[15] J. Hendrickx and J. Tsitsiklis, "A new condition for convergence in continuous-time consensus seeking systems," in *IEE 50th CDC-ECC conference*, Dec. 2011, pp. 5070 –5075.

[16] E. Seneta, *Non-negative Matrices and Markov Chains (Springer Series in Statistics)*, 2nd ed. Springer, January 2006.

Contents



RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399